Ant colony optimization algorithm for solving multidimensional knapsack problem

Michał Reclik

Wrocław University of Technology,
Faculty of Fundamental Problems of Technology,
Department of Algorithmic Information Technologies

June 2020

1 Abstract

An ant colony optimization, or ACO for short, is a metaheuristic search algorithm best suited for solving discrete optimization problems where solutions can be represented as paths in a graph. ACO, as name suggest, is inspired by real-life ants behaviour which follow scent of a pheromone left by other ants. This paper aims to properly explain two chosen implementations of the mentioned algorithm in the np-hard multidimensional knapsack problem.

2 Introduction

The multidimensional knapsack problem (later referred to as MKP) is a very popular combinatorial problem with countless applications across all industries i.e. capital budgeting problem, network selection for mobile nodes or internet download manager. Objective is to maximize the profit function based on the chosen items while satisfying capacity constraints for each dimension in the considered scenario. This can be formulated formally as

$$maximize \quad f(x) = \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$satisfy \quad \sum_{j=1}^{n} r_{,ij} x_j \leq b_i, \quad i = 1, ..., m \tag{2}$$

$$x \in \{0, 1\}, \quad j = 1, ..., n \tag{3}$$

Now consider the MKP modeled as a graph. Let $o_j, j = 1, ..., n$ be collectable objects. Now, let $G(V, E)$ be a complete, undirected graph with n vertices representing each o_j object.

With the problem properly defined the ACO is applicable.

3 ACO and multidimensional knapsack problem

Developed in 1990s by M. Dorigo, ACO relies on artificial ants which build their path on the graph by choosing the next object with probability dictated by pheromone left by other ants.

3.1 Probability of traversal

Amount of pheromone luring ants to traverse an edge of the graph (o_i, o_j) will be denoted as $\tau(o_i, o_j)$. Contrary to real-life path finding, the MKP does not require solution to be considered in order, thus the pheromone intensity prompting an ant to choose o_j can be calculated in a following way[1]

$$\tau_{S_k}(o_i) = \sum_{o_j \in S_k} \tau(o_i, o_j) \tag{4}$$

Where S_k is the currently build solution

Beside the pheromone factor this implementation of ACO also considers a metaheuristic factor, which in this case represents how profitable the object o_j is relative to the percentage of remaining dimensional resources it consumes (often referred to as 'tightness' ratio). It acts as a counterweight for the pheromone factor so that ants do not converge too fast. Formally denoted as

$$\eta_{S_k}(o_j) = \frac{p_j}{h_{S_k}(j)} \tag{5}$$

Where $h_{S_k}(o_j)$ is the mentioned tightness ratio.

$$h_{S_k}(o_j) = \sum_{i=1}^{m} \frac{r_{i,j}}{d_{S_k}(i)} \tag{6}$$

Where $d_{S_k}(i)$ is the remaining capacity in $i-th$ dimension. More specifically, let $c_{S_k}(i) = \sum_{g \in S_k} r_{i,g}$ be the cost of $k-th$ solution in $i-th$ dimension. Then $d_{S_k}(i) = b(i) - c_{S_k}(i)$

Now we can build the probability equation integrating the formulas mentioned above

$$p_{o_i} = \frac{[\tau_{S_k}(o_i)]^\alpha \cdot [\eta_{S_k}(o_i)]^\beta}{\sum_{o_j \in allowed_{S_k}} [\tau_{S_k}(o_j)]^\alpha \cdot [\eta_{S_k}(o_j)]^\beta} \tag{7}$$

Where $allowed_{S_k}$ is a set of unpicked items in $k-th$ solution that do not violate dimensional constraints when added. Items that do violate given constraints have zero probability of being selected.

Starting values are initialised with the beginning of the program to default non-zero positive values.

3.2 Pheromone evaporation and emission

Simulated pheromone is, just like the real one, prone to the process of evaporation, which strengthens relative importance of recently chosen objects.

3.2.1 Ant-Knapsack pheromone model [1]

In the standard ACU model it is possible to have just one ant that deposits the pheromone. The chosen ant S_k is the best of the current generation (best MKP profit function) and lays down $1/(1+L(S_{best})-L(S_k))$ pheromone on each edge of the constructed complete graph S_k. This value is inversely proportional to the difference in quality between S_{best} and S_k to emphasise on better solutions. Note that if we were to use equation 4 to calculate the pheromone factor it can done incrementally. More specifically, if o_i is the first chosen vortex then we initialize $\tau_{S_k}(o_j)$ for each item $o_j \in allowed$ to $\tau(o_i, o_j)$. Then each time item o_l is chosen $\tau_{S_k}(o_j)$ is simply incremented by $\tau(o_l, o_j)$.
The rate of evaporation is dictated by a program parameter ρ, such that $0 < \rho < 1$. The evaporation happens after every ant has finished building the solution and is executed by multiplying the pheromone levels at each edge by $(1-\rho)$ [1]. Also, to keep the pheromone values in reasonable boundaries that support exploration, we define τ_{min} and τ_{max}. After laying the pheromone, our updated values are checked and normalised to τ_{min} or τ_{max} if exceed given boundary.

3.2.2 ACOMPD pheromone modification [2]

The deployed pheromone on the graph vortex as mentioned by Junzhong Ji uses distance association (DA for short) which is touched upon in the next paragraph. It is equal to[2]

$$\Delta\tau(o_i) = \sum_{S_k \in S} \Delta\tau_{S_k}(o_i) \tag{8}$$

$$\Delta\tau_{S_k}(o_i) = \begin{cases} Q \cdot L(S_k), & o_i \in S_k \\ 0, & otherwise \end{cases} \tag{9}$$

Where $L(S_k)$ is the MKP function value for solution S_k and Q is a constant parameter.

Now for the diffusion model that incorporates DA

$$\Delta\tau(o_i, o_j) = \begin{cases} \frac{1}{N+1} \cdot \frac{\Delta\tau(o_i)}{d_r(o_i, o_j)}, & d_r(o_i, o_j) < 1 \\ 0, & otherwise \end{cases} \tag{10}$$

Where $d_r(o_i, o_j)$ is the distance association and N is the number of solutions. Now we can formulate the final equation for pheromone updating

$$\tau(o_i) = \tau(o_i) + \sum_r \Delta\tau(o_r, o_i) \tag{11}$$

Where o_r is an object in out data base with connections to o_i.

4 ACOMPD modifications

Aside from the difference in pheromone calculation the more advanced ACOMPD algorithm applies other modifications to our standard ACO model to help reduce time complexity and give prominence to more valuable solutions.

4.1 Association distance by Top-k best solutions

Being a relatively simple solution to reduce time complexity, it is widely used i.e. in search engines. More specifically let D be a database with our solutions S. Then searching only k top solutions by the MKP function we calculate the frequency f, which tells us how many times given pair (o_i, o_j) appears in D. Now to calculate the association distance between two items

$$d_r(o_i, o_j) = \frac{1}{f+1} \tag{12}$$

4.2 Convergent mutation

Despite previous modifications helping to improve the global search aspect of MKP, current algorithm needs a huge number of iterations to converge to optimal solution. This can be fixed by semi-controlled mutations that happens after fixed number of iterations, let be M. Then if the best solution S_{best} and current best solution $\S_{kbest}$ are identical then an item in S_{kbest} is replaced with an item from the *allowed* set. If this process produce a better solution that does not violate dimensional constraints then the mutation is accepted.

5 Ant-Knapsack algorithm

Finally we will focus on the basic implementation of the Ant-Knapsack to properly show problems this algorithms faces.

```
In: Objects values and costs, dimensional constraints, max time;
initialization: initialize pheromone intensity on each edge to c;
while time left do
    for S_k ∈ S do
        S_k = ∅
        allowed = x
        while allowed ≠ ∅ do
            choose random o_j from allowed
            with probability p_j add it to S_k and remove from allowed
            update allowed so that none added element would violate
              constraints
        end
    end
    Update pheromone levels
    Values that exceed accepted boundaries normalise to τ_min or τ_max
end
```

6 Empirical study of Ant-Knapsack

While testing the ACO on a relatively small dataset it is easier to see and work on the program parameters, which are in our case responsible for

- α - pheromone factor relative importance
- β - heuristic factor relative importance
- ρ - pheromone evaporation pace
- N - number of ants in a colony

The goal is to balance the convergence speed with exploration aspect. Furthermore the evaporation has to be strong enough not to let too many objects reach upper pheromone limit, and low enough to let the pheromone persist for some time without reaching lower limit.

With the Ant-Knapsack implemented and tested on a small dataset (50 objects, 2 dimensions) with $\alpha = 1$, $\beta = 5$, $N = 10$, $\rho = 0.02$ algorithms best solution was usually found within first 10 cycles. However, to observe a meaningful decrease in the standard deviation between results, about 20 cycles were needed. To enhance the convergence speed, one can intuitively increase α, but with such small dataset and limited number of cycles this action can have a negative impact on the convergence if one does not increase the ρ parameter to faster emphasise the pheromone differences. With regard to the above, parameter configuration $\alpha = 2$, $\beta = 5$, $\rho = 0.08$ managed to struck a balance between intensification and diversification.

7 Comparisons

While comparing the I. Alaya's Ant-Knapsack with the modified version ACOMPD we can observe huge differences in the overall convergence speed. Test shown below were conducted with 100 objects to choose and 5 dimensional constraints. The Ant-Knapsack was tuned with respective parameters: $\alpha = 1$, $\beta = 5$, $\rho = 0.01$, $N = 30$, $\tau_{min} = 0.01$, $\tau_{max} = 6$ and the ACOMPD: $\alpha = 1$, $\beta = 5$, $\rho = 0.3$, $k = 20$ (for Top_k), $M = 5$ and $N = n$ (equal to number of objects).

Known Optimum	Ant-Knapsack	AK iterations	ACOMPD	ACOMPD iterations
24381	24381	522	24381	11
24274	24274	469	24274	12
23551	23551	483	23551	10
23534	23534	500	23534	35
23991	23991	589	23991	19

Though both algorithms managed to find the known optima, there is an order-of-magnitude-level difference in the number of average iterations to find the desirable value.

8 Conclusions

It is clear that the more basic Ant-Knapsack[1], even with fine tuned parameters in I. Alaya's research paper, struggles with convergence speed. This goes to show what impact modifications in the ACO algorithm, such as those proposed by Junzhong Ji in the ACOMPD[2] model, can have. Mutation strategy is likely to be the key in reducing time taken to converge while modifications such as pheromone diffusion model based on the distance association help in maintaining exploration and finally the Top_k search query reduce time taken by one cycle in a more complicated model. Overall, the Ant-Knapsack[1] is a simpler, is easier to implement and further modify, however it could not achieve the same level of convergence speed as the more advanced (and complicated) ACOMPD[2] model.

References

[1] Khaled Ghedira In'es Alaya, Christine Solnon. *ANT ALGORITHM FOR THE MULTIDIMENSIONAL KNAPSACK PROBLEM.*

[2] Chunnian Liu Xuejing Liu Ning Zhong Junzhong Ji, Zhen Huang. *An Ant Colony Optimization Algorithm for Solving the Multidimensional Knapsack Problems.*